SUPPORTING STRUCTURES

PROGRAM STRUCTURES

- SPMD
- MASTER/WORKER
- LOOP PARALLELISM
- FORK/JOIN

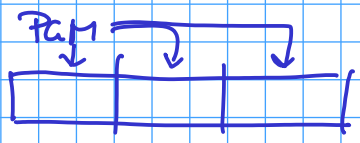
DATA STRUCTURES

SHARED DATA

SHARED QUEUE

DISTRIBUTED ARRAY

SPMD (Single Program Multiple data)



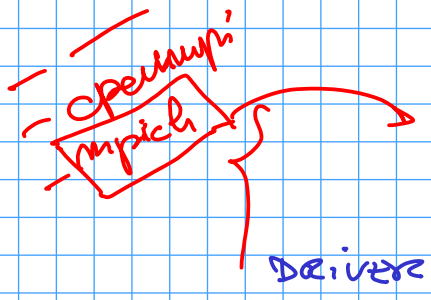
MPI — message passing interface

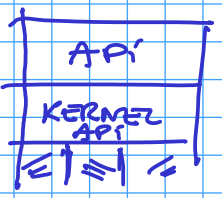
MPI_init(—);

(PGM_id) =

MPI_Finalize();

LIBRARY



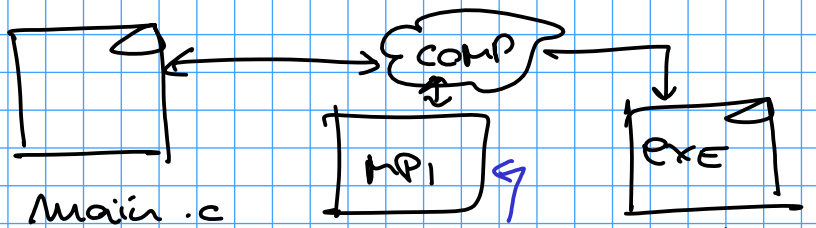


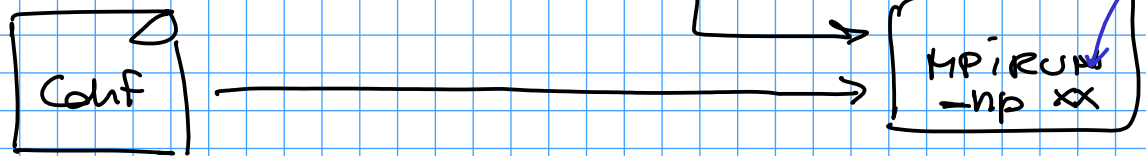
≈ 200

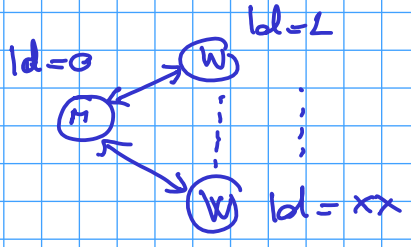
≈ 10

set of operations:
- init
- finalize
- send/receive
- collective comm { broadcast, multicast
- one sided communications / RDMA

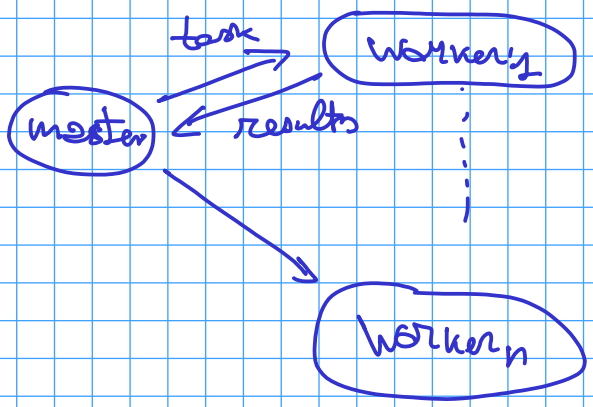
RUN/DEPLOY process







MASTER / WORKER



schema 1

$\forall w$ compute the same kind of tasks

$w : f$

schema 2

$\forall w$ may compute 1 of a set of tasks depending on master requests

$w : \{f_1 f_2 \dots f_n\}$

schema 3

$\forall w_i$ compute f_i

(M/W with specialized / dedicated workers)

worker :

```
while (true) {
    receive (task)
    results = compute(task)
    send (results)
}
```

try to overlap comm & comp !

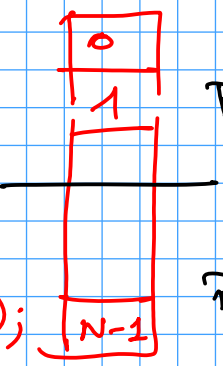
LOOP PARALLELISM PATTERN

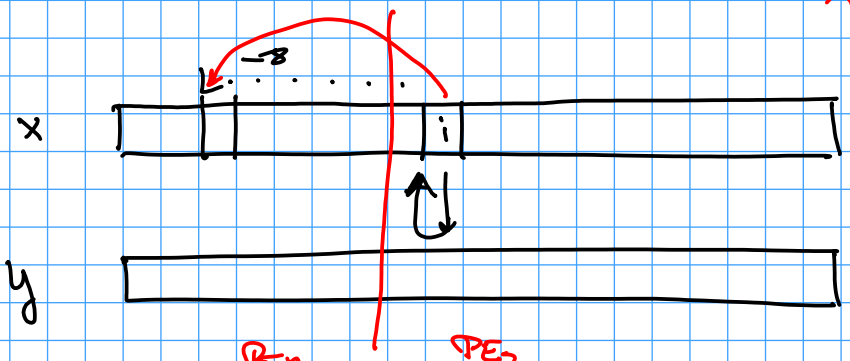
Iterative comp

for( ) { body }
(while( ) { body })

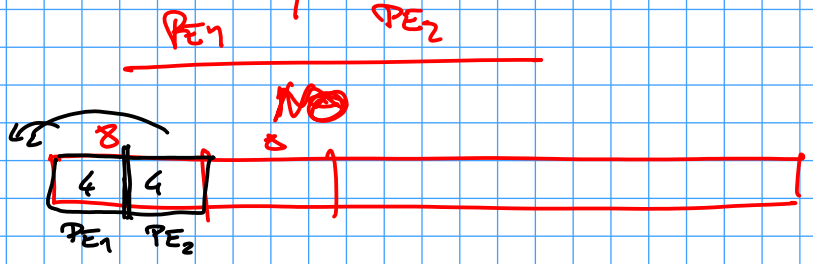
```
for(i=0; i<N; i++) {
    x[i] = f(y[i]);
}
```

```
x[0] = f(y[0]);
x[1] = f(y[1]);
   ⋮
x[N-1] = f(y[N-2]);
```






```
for(          ) {
  y[i] = f(x[i], x[i-8])
}
```



x 00123123

```
for(i=0 ; i<6 ; i++)
    x[i] = x[i] + x[i-2]
```

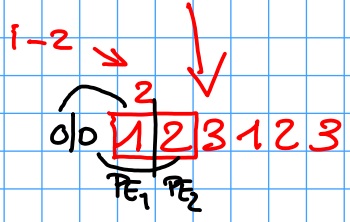
x[6] x[5]

00123123
i=0 00 1 23123
i=1 001 2 3123
i=2 00 1 2 4 123
3
6
9
124369

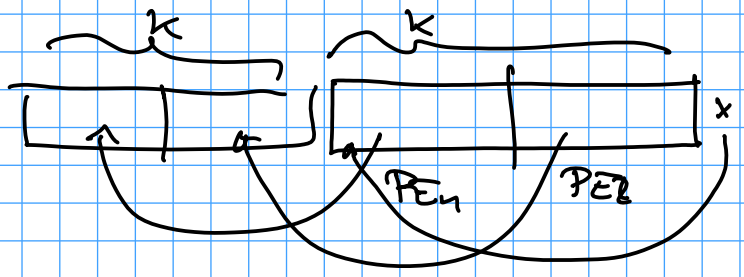
missing elements i=0 x[0]+x[-2]
↑
∅!!

y[i] = x[i] + x[i-2]

123123
124354



$x[i] = f(x[i], x[i-k])$



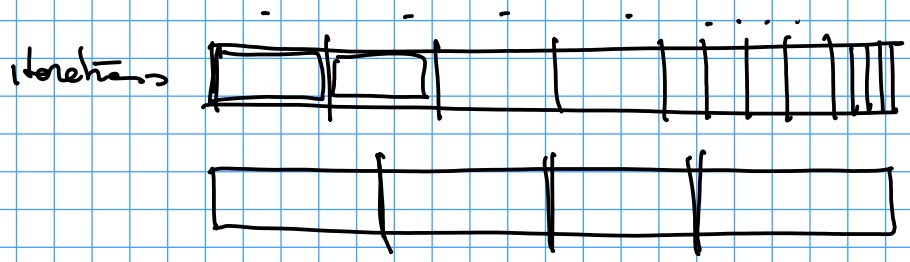
$k=2$ ❗

$x_i \quad x_{i-1}$

The way you partition the iterations matters (is important)

```
#pragma omp parallel for
for(i=0; i<N; i++) {
    ≡
}
```

independent iterations ❗

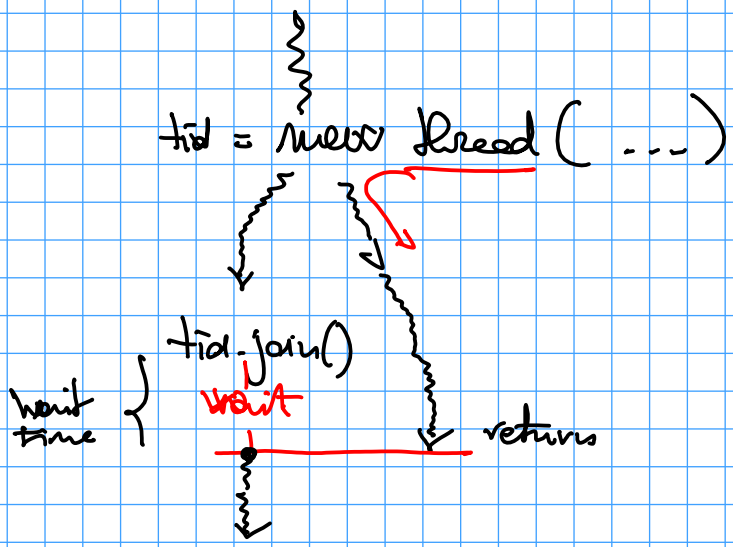


+ job stealing

Fork / Join

generates a parallel activity

await termination of a parallel activity



Cilk Plus

Cilk +

cilk_spawn
cilk_sync

-fcilkplus

added an equivalent of the parallel for

SHARED DATA

manage the accesses

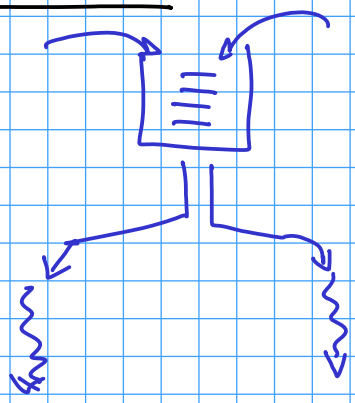
↳ patterns for accesses

readers / writers

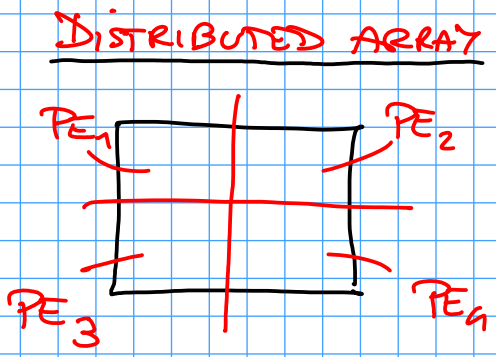
↳ synchronizations

↳ reducing the size of critical sections

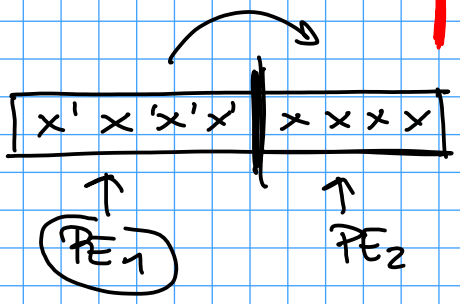
SHARED QUEUE



DISTRIBUTED ARRAY

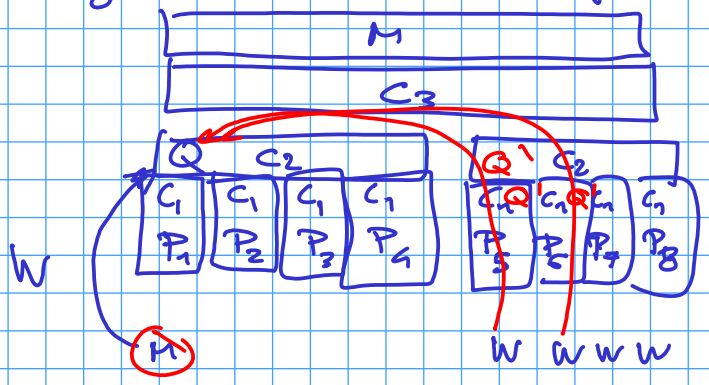


Cache



"false sharing"

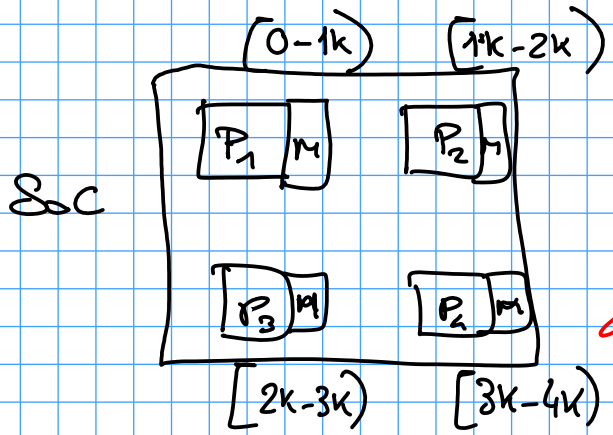
e.g. master worker implementation

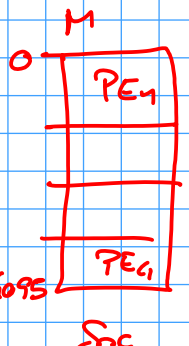


alignment

cache

$A \times B \approx A \times (B^T)$





P_1 runs a program

uses m[0] ÷ m[1023]

to communicate to P_4

write bee m[4095]